ATTORNEY DOCKET NO.
AUS920030713US1 (IBM2804000)

PATENT APPLICATION
SERIAL NO. 10/733,840

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of: Chow et al.	§	Confirmation No.: 2185
	§	
Serial Number: 10/733,840	§	Group Art Unit: 2195
	§	
Filed: December 11, 2003	§	Examiner: Camquy Truong
	§	
For: METHOD FOR HIDING LATENCY IN A TASK-BASED LIBRARY FRAMEWORK FOR A MULTIPROCESSOR ENVIRONMENT	§	

CERTIFICATE OF EFS-WEB TRANSMISSION

Pursuant to 37 C.F.R. § 1.8, I hereby certify that this correspondence is being electronically transmitted to the United States Patent and Trademark Office at www.uspto.gov

on: June 16, 2008

/Bradley D. Ellis/
Bradley D. Ellis

Commissioner of Patents and Trademarks
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPLICANTS' APPEAL BRIEF

Applicant-inventors ("Applicants") and assignee International Business Machines Corporation respectfully submit the present brief in support of the patentability of the claims of the above-referenced application.

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, of Armonk, New York, assignee of the interests in the invention from the named inventors.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF CLAIMS

Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 are pending. Of these, Claims 1, 9, and 15 are independent Claims. Applicants have previously cancelled Claims 3-4, 7-8, 11, 14, 17-18, and 21-22. Applicants appeal the Examiner's rejections of Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 under 35 U.S.C. §103(a).

IV. STATUS OF AMENDMENTS

The Claims stand as amended in the Response, filed November 2, 2007, to an Office Action of August 3, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

A multiprocessor system subdivides programs into tasks, assigned among the multiple processors, which together execute a program faster than a single processor of the same speed, because the multiple processors work simultaneously on the program. *See* Application, Page 1, Lines 10-14. In traditional methods of load balancing, which is apportions the tasks among the processors, each processor has a queue of tasks and a central task-distributor assigns each new task on arrival to a particular processor's queue. *See* Application, Page 1, Lines 16-21.

A typical central distributor tries to predict how long each processor requires to complete the tasks in its queue, using methods such as round-robin, random assignment, and assessment of how busy the processors are. *See* Application, Page 1, Lines 21-25. But the distributor's

assessment is not always accurate, and as a result, some processors sometimes have long queues of tasks while others are idle, resulting in program execution delay. *See* Application, Page 1, Lines 25-29. Moreover, the distributor itself can become overburdened, which also introduces program execution delay. *See* Application, Page 1, Line 29 to Page 2, Line 3.

The present invention, defined in the pending Claims, improves contemporary load balancing techniques by introducing a central queue, from which processors can self-assign tasks. *See* Application, Page 3, Lines 24-31. The system memory includes a task queue (called a "library task queue"), and each processor (called a "library processor") has access to the library task queue. *See* Application, Page 3, Lines 24-27. The system kernel divides incoming processing tasks into subtasks and puts the subtasks into the library task queue. *See* Application, Page 3, Lines 27-30. The library processors fetch the subtasks from the library task queue. *See* Application, Page 3, Lines 30-31.

The Claims embody the invention as follows, shown with illustrative citations to page and line numbers in the Original Application designated in curved braces ("{}"):

1. (Previously Presented) A method for load balancing in a tightly-coupled multiprocessor computer system comprising the steps of:

dividing a task into a plurality of subtasks;{Page 4, lines 22-23}

placing the plurality of subtasks into a centralized task queue; {Page 3, lines 27-30}

distributing the plurality of subtasks in the centralized task queue to a plurality of library processors, wherein each library processor comprises exactly two task buffers; {Page 4, lines 1-2, 22-23; Page 6, lines 7-17}

wherein at least one subtask from the plurality of subtasks in the centralized task queue is distributed to at least one of the plurality of library processors when the library processor has at least one empty task buffer; {Page 4, lines 2-6} and

wherein distributing a subtask from the plurality of tasks in the centralized task queue to the one of the plurality of library processors comprises the one of the plurality of library processors fetching the subtask from the centralized task queue {Page 4, lines 2-3}.

9. (Previously Presented) A system for load balancing in a tightly-coupled multiprocessor computer system comprising

a system kernel configured to receive a task and to divide the received task into a plurality of subtasks; {Page 4, lines 22-23}

a library task queue coupled to the system kernel; {Page 3, lines 24-26}

a plurality of library processors coupled to the library task queue, wherein each of the plurality of library processors comprises exactly two task buffers; {Page 4, lines 1-2, 22-23; Page 6, lines 7-17}

wherein the system kernel is configured to place the plurality of subtasks into the library task queue; {Page 3, lines 27-30} and

wherein each of the plurality of library processors is configured to fetch at least one subtask of the plurality of subtasks from the library task queue. {Page 4, lines 2-3}

15. (Previously Presented) A computer program product for load balancing in a tightly-coupled multiprocessor computer system, the computer program product having a medium with a computer program embodied thereon, the computer program comprising:

computer code for dividing a task into a plurality of subtasks; {Page 4, lines 22-23}

computer code for placing the plurality of subtasks into a centralized task queue; {Page 3, lines 27-30}

computer code for distributing the plurality of subtasks in the centralized task queue to a plurality of library processors, wherein each library processor comprises exactly two task buffers; {Page 4, lines 1-2, 22-23; Page 6, lines 7-17}

wherein a subtask from the plurality of subtasks in the centralized task queue is distributed to one of the plurality of library processors when the library processor has at least one empty task buffer; {Page 4, lines 2-6} and

wherein distributing a subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors comprises the one of the plurality of library processors fetching the subtask from the centralized task queue. {Page 4, lines 2-3}

VI. GROUNDS OF REJECTION TO BE REVIEWED

Whether Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 are patentable over Sundaresan (US 6,289,369)("Sundaresan") in view of Willen et al. (US 7, 159, 221)("Willen").

VII. ARGUMENT

A. Grouping of Claims

Claims 1, 9, and 15 are independent. For purposes of this appeal, Applicants consider each of the independent Claims, and their respective dependent Claims, as separate groups. Thus, the groups of Claims are {1-2, 5-6}, {9-10, 12-13}, and {15-16, 19-20}.

B. Summary of Pertinent Prosecution

Applicants filed the present application on December 11, 2003, with 22 claims.

The Examiner mailed the first substantive Office Action, the "First Action," on August 3, 2007. In addition to some minor Section 112 rejections, the Examiner rejected the claims under Section 103(a) as anticipated by Sundaresan in view of Willen and further in view of Baumberger (US Patent Application Pub. No. 2005/0102671).

Applicants responded to the First Action on November 2, 2007 ("First Response"), amending Claims 1-2, 5-6, 9-10, 12-13, 15-16, 19-20. Applicants cancelled Claims 3-4, 7-8, 11, 14, 17-18, and 21-22.

The Examiner mailed the Final Action under appeal ("Final Action") on January 15, 2008. In the Final Action, the Examiner rejected the pending claims under Section 103(a) over Sundaresan in view of Willen. This appeal followed.

C. The Examiner's Rejections Were Procedurally and Factually in Error

1. The Form and Content of the Examiner's Rejections under Section 103 Were Improper and Insufficient

a. Legal Requirements for an Obviousness Rejection

The obligation of the examiner to produce reasoning and evidence in support of obviousness is clearly defined at M.P.E.P. §2142:

> The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness. If the examiner does not produce a *prima facie* case, the applicant is under no obligation to submit evidence of nonobviousness.

M.P.E.P. §2143 sets out the three basic criteria that a patent examiner must satisfy to establish a *prima facie* case of obviousness:

1. some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings;
2. a reasonable expectation of success; and
3. the teaching or suggestion of all the claim limitations by the prior art reference (or references when combined).

It follows that in the absence of such a *prima facie* showing of obviousness by the Examiner (assuming there are no objections or other grounds for rejection), an applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443 (Fed. Cir. 1992). Thus, in order to support an obviousness rejection, the Examiner is obliged to produce evidence compelling a conclusion that each of the three aforementioned basic criteria has been met.

b. The Examiner's Stated Grounds Were Insufficient

As described above, the Final Action rejects Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 under 35 U.S.C. §103(a) as allegedly unpatentable over Sundaresan in view of Willen. Applicants respectfully submit that these rejections are in error and should be withdrawn.

The Examiner must meet at least three statutory requirements to bring a sufficient *prima facie* case of obviousness: suggestion or motivation to combine, a reasonable expectation of success, and the teaching or suggestion of all the claim limitations. *See* M.P.E.P. §2143. Applicants respectfully submit that the Examiner has at least failed to demonstrate the teaching or suggestion of all the claim limitations, as described in more detail below.

Specifically, independent Claims 1 and 15 recite "wherein each library processor comprises exactly two task buffers". Independent Claim 9 recites, "wherein each of the plurality of library processors comprises exactly two task buffers". In the First Action, the Examiner admits that "Sundaresan and Willen do not explicitly teach the one of the plurality of library processors has exactly two task buffers." First Action, Page 7. To supply this missing element,

the Examiner offered Baumberger, asserting that "Baumberger teaches . . . the one of the plurality of library processors has exactly two task buffers (paragraph 16, lines 1-6)." First Action, Page 7. Applicants successfully overcame this argument in the First Response. *See* First Response, Page 8.

In the Final Action, the Examiner now relies exclusively on Sundaresan and Willen to disclose "exactly two task buffers" as recited in the claims, notwithstanding the Examiner's prior admission that Willen does not disclose this element. The Examiner now states, "Sundaresan does not explicitly teach each library processor comprise[s] exactly two task buffers" and that "Willen teaches each library processor comprise[s] exactly two task buffers (each processor in the system has its own queues (SQ0 – SQn-1), col. 14, lines 56-57." Final Action, Page 3, Para. 6. Applicants respectfully submit that this assertion is inaccurate on its face. As the Examiner states, Willen teaches "0" to "n-1" queues, which directly contradicts teaching "exactly two" queues. *See, e.g.*, Willen, Figure 2A, Figure 2B. As such, Willen cannot be said to teach "exactly two task buffers" as it expressly contradicts such teaching.

The Examiner also asserts that "Applicant has not disclosed that the exactly two task buffers provides an advantage, or solves a stated problem." Final Action, Page 6. Applicants respectfully disagree. The Specification recites:

> Since a library processor 108, 110, 112, 114, or 116 fetches tasks only when there is at most one task in the buffers, the load on a library processor is never more than two tasks, one of which is executing. As a result, the load is evenly balanced.

Original Application, Page 6, lines 9-14. Thus, Applicants have expressly disclosed an advantage of exactly two task buffers and the Examiner's argument fails. Accordingly, the Examiner's proposed combination of Sundaresan and Willen fails to teach the element that each library processor comprises exactly two task buffers.

But the Examiner's proposed combination also fails to teach other claim elements. For example, the Examiner also admits that "Sundaresan does not explicitly teach . . . one of the plurality of library processors fetching the subtask from the centralized task queue." Final Action, Page 3. The Examiner offers Willen to provide this missing element, asserting that Willen teaches "one of the plurality of library processors fetching the subtask from the centralized task queue (idle processor steal[s] work *from the queues of processors that are busier* on average, col. 11, lines 42-45; col. 15, lines 1-8; col. 16, lines 30-39)." Final Action, Pages 3-4. Once again, Applicants respectfully submit that the Examiner's argument is plainly wrong on its face.

Specifically, the Examiner states that Sundaresan does not teach the library processors retrieving tasks from the *centralized task queue*, and offers Willen to show processors stealing from *each other's queues*, which are plainly not a centralized task queue. Willen clearly teaches away from a centralized task queue, instead teaching idle processors "stealing" work from busier processors. *See, e.g.*, Willen, col. 11, lines 42-45. Accordingly, the Examiner's proposed combination, as described by the Examiner, not only fails to teach the elements as recited in the Claims, but affirmatively teaches away from the elements as recited in the Claims.

Moreover, the Examiner also states that "it would have been obvious to combine . . . fetching the subtask from the centralized task queue *as taught by Willen* to the invention of Sundaresan . . ." Final Action, Page 4, Para. 7 (emphasis added). As described above, the invention "as taught by Willen" expressly contradicts the invention as recited in the claims. Therefore, by the Examiner's own reasoning, it would not have been obvious to combine "fetching the subtask from the centralized task queue" *as recited in the Claims*, with the teaching of Sundaresan, which underscores the novelty embodied in the pending Claims.

For at least the above reasons, Applicants respectfully submit that the Examiner's proposed combination fails to teach each and every element in the pending Claims, and therefore fails as a *prima facie* case. Because the Examiner's *prima facie* case fails, the rejections based on that case fail, and must be withdrawn.

2. Conclusion

Accordingly, Applicants respectfully submit that the Examiner's stated grounds are insufficient to maintain the Final Rejections. Applicants therefore respectfully request that the Final Rejections be withdrawn and that Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 be allowed.

VIII. CLAIMS APPENDIX

See Attached.

IX. EVIDENCE APPENDIX

NONE.

X. RELATED PROCCEDINGS APPENDIX

NONE.

XI. CONCLUSION

For the foregoing reasons, Applicants respectfully submit that the Final Rejections of Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 under 35 U.S.C. §103(a) are improper and should be reversed. Applicants respectfully request that the rejections of Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 be withdrawn and that Claims 1-2, 5-6, 9-10, 12-13, 15-16, and 19-20 be allowed.

Applicants hereby authorize the Director to charge the required fee for the filing of this Appeal Brief to Deposit Account No. 09-0447 of IBM Corporation. Applicants do not believe that any other fees are due; however, in the event that any other fees are due, the Director is hereby authorized to charge any required fees due (other than issue fees), and to credit any overpayment made, in connection with the filing of this paper to Deposit Account No. 09-0447 of IBM Corporation.

Respectfully submitted,

CARR LLP

/Gregory W. Carr/
Gregory W. Carr
Reg. No. 31,093

Dated: June 16, 2008
CARR LLP
670 Founders Square
900 Jackson Street
Dallas, Texas 75202
Telephone: (214) 760-3030
Fax: (214) 760-3003

VIII – APPENDIX – CLAIMS ON APPEAL

1. (Previously Presented) A method for load balancing in a tightly-coupled multiprocessor computer system comprising the steps of:

dividing a task into a plurality of subtasks;

placing the plurality of subtasks into a centralized task queue;

distributing the plurality of subtasks in the centralized task queue to a plurality of library processors, wherein each library processor comprises exactly two task buffers;

wherein at least one subtask from the plurality of subtasks in the centralized task queue is distributed to at least one of the plurality of library processors when the library processor has at least one empty task buffer; and

wherein distributing a subtask from the plurality of tasks in the centralized task queue to the one of the plurality of library processors comprises the one of the plurality of library processors fetching the subtask from the centralized task queue.

2. (Previously Presented) The method of claim 1, further comprising distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors when the one of the plurality of library processors has one or two empty task buffers.

3. – 4. (Cancelled).

5. (Previously Presented) The method of claim 1, further comprising distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors by the one of the plurality of library processors fetching the subtask from the centralized task queue when the load of the one of a plurality of library processors is zero or one subtasks.

6. (Previously Presented) The method of claim 1, further comprising distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors by the one of the plurality of library processors fetching the subtask [[it]] from the centralized task queue when the load of the one of a plurality of library processors is zero subtasks.

7. – 8. (Cancelled).

9. (Previously Presented) A system for load balancing in a tightly-coupled multiprocessor computer system comprising

a system kernel configured to receive a task and to divide the received task into a plurality of subtasks;

a library task queue coupled to the system kernel;

a plurality of library processors coupled to the library task queue, wherein each of the plurality of library processors comprises exactly two task buffers;

wherein the system kernel is configured to place the plurality of subtasks into the library task queue; and

wherein each of the plurality of library processors is configured to fetch at least one subtask of the plurality of subtasks from the library task queue.

10. (Previously Presented) The system of Claim 9, wherein each of the plurality of library processors further is further configured to fetch a subtask from the library task queue when that library processor has at least one empty task buffer.

11. (Cancelled).

12. (Previously Presented) The system of Claim 9, wherein the system kernel is comprised of a single processor.

13. (Previously Presented) The system of Claim 9, wherein the system kernel is comprised of a plurality of processors.

14. (Cancelled).

15. (Previously Presented) A computer program product for load balancing in a tightly-coupled multiprocessor computer system, the computer program product having a medium with a computer program embodied thereon, the computer program comprising:

computer code for dividing a task into a plurality of subtasks;

computer code for placing the plurality of subtasks into a centralized task queue;

computer code for distributing the plurality of subtasks in the centralized task queue to a plurality of library processors, wherein each library processor comprises exactly two task buffers;

wherein a subtask from the plurality of subtasks in the centralized task queue is distributed to one of the plurality of library processors when the library processor has at least one empty task buffer; and

wherein distributing a subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors comprises the one of the plurality of library processors fetching the subtask from the centralized task queue.

16. (Previously Presented) The computer program product of Claim 15, further comprising computer code for distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors when the one of the plurality of library processors has one or two empty task buffers.

17. – 18. (Cancelled).

19. (Previously Presented) The computer program code of Claim 15, further comprising computer code for distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors by the one of the plurality of library processors fetching the subtask from the centralized task queue when the load of the one of a plurality of library processors is zero or one subtasks.

20. (Previously Presented) The computer program code of Claim 15, further comprising computer code for distributing the subtask from the plurality of subtasks in the centralized task queue to the one of the plurality of library processors by the one of the plurality of library processors fetching the subtask from the centralized task queue when the load of the one of a plurality of library processors is zero subtasks.

21. – 22. (Cancelled).